УДК 004.58

Анализ алгоритмов обучения коллаборативных рекомендательных систем

© Д.Е. Королева, М.В. Филиппов

МГТУ им. Н.Э. Баумана, Москва, 105005, Россия

В статье рассмотрены алгоритмы коллаборативной фильтрации, используемые в рекомендательных системах. Проведен сравнительный анализ данных алгоритмов с точки зрения критериев точности полученных результатов и быстродействия. Даны рекомендации по их использованию в конкретных случаях.

Ключевые слова: *рекомендательные системы, коллаборативная фильтрация, предикат, условная вероятность, кластеризация, коэффициент корреляции.*

Введение. Рекомендательные системы — программные средства, которые пытаются предсказать какие объекты (фильмы, музыка, книги, новости, веб-сайты и т. д.) будут интересны пользователю, если имеется определенная информация о его предпочтениях.

Существуют две основные стратегии создания рекомендательных систем: фильтрация содержимого и коллаборативная фильтрация [1,2].

При фильтрации содержимого создаются профили пользователей и объектов. Профили пользователей могут включать демографическую информацию или ответы на определенный набор вопросов. Профили объектов могут включать названия жанров, имена актеров, исполнителей и т. п. в зависимости от типа объекта.

При коллаборативной фильтрации [3] используется информация о поведении пользователей в прошлом, например о покупках или оценках. В этом случае не имеет значения с какими типами объектов ведется работа, но могут учитываться неявные характеристики, которые сложно было бы учесть при создании профиля. Основная проблема этого типа рекомендательных систем — так называемый «холодный старт», а именно, отсутствие данных о недавно появившихся в системе пользователях или объектах.

Рассмотрим некоторые широко известные рекомендательные системы:

• *Amazon* — один из лидеров подобных систем. *Amazon* рекомендует книги и другие товары, основываясь на том, что вы покупали, что просматривали, какие рейтинги ставили и какие оставляли отзывы;

• *Last.fm* и *Pandora* рекомендуют музыку. Они придерживаются разных стратегий рекомендации: *Last.fm* использует, кроме собственно рейтингов других пользователей, исключительно «внешние» дан-

ные о музыке — автор, стиль, дата, тэги и т. п. *Pandora* основывается на «содержании» музыкальной композиции, используя очень интересную идею — Music Genome Project, в котором профессиональные музыканты анализируют композицию по нескольким сотням атрибутов (в России *Pandora* сейчас недоступна);

• *Google*, *Yahoo!*, *Яндекс* — можно сказать, что они тоже рекомендуют пользователям сайты, но на самом деле это другие системы: поисковики пытаются предсказать, насколько данный документ отвечает данному запросу, а рекомендатели — какой рейтинг данный пользователь поставит данному продукту. Несколько ближе к нашей задаче проблема того, какую рекламу показывать пользователю (*AdSense*, *Яндекс.Директ* и т. д.) — здесь нужно «порекомендовать» те из них, которые, скорее всего, вызовут положительную реакцию. Однако у ведущих поисковиков есть масса побочных проектов, основанных на рекомендательных системах — например, *Yahoo! Music*.

Большинство коллаборативных рекомендательных систем использует алгоритм Байеса или алгоритм SVD (или SVD++). Но оба эти алгоритма требуют большой обучающей выборки. В настоящей работе будет проведен их сравнительный анализ, а также предложен альтернативный алгоритм, не требующий больших обучающих выборок. Сравнение проводилось по критериям правильности рекомендации и быстродействия.

Описание алгоритмов. ***Алгоритм Байеса.*** Теорема Байеса [4] — одна из основных теорем элементарной теории вероятностей, которая определяет вероятность наступления события в условиях, когда на основе наблюдений известна лишь некоторая частичная информация о событиях.

Условная вероятность события x при условии события y обозначается $p(x|y)$. Согласно теории вероятностей

$$p(x\big|y) = \frac{p(x,y)}{p(y)},$$

где $p(x,y)$ — это совместная вероятность событий x и y, а $p(x)$ и $p(y)$ — вероятности каждого события по отдельности. Таким образом, совместную вероятность можно выразить двумя способами:

$$p(x,y) = p(x\big|y)p(y) = p(y\big|x)p(x).$$

По теореме Байеса условная вероятность $p(x|y)$ определяется следующим выражением [4]:

$$p(x\big|y)p = \frac{p(y\big|x)p(x)}{p(y)}.$$

В контексте задачи машинного обучения y — это исходные данные, т. е. известная информация, а x — параметры модели, которые мы хотим обучить. Например, представим данные как рейтинги, которые ставили пользователи продуктам, а параметры модели — факторы, которые мы обучаем для пользователей и продуктов.

Каждая из вероятностей тоже имеет свой смысл. $p(x|y)$ — распределение вероятностей параметров модели после того, как были учтены исходные данные. Эта вероятность также называется апостериорной вероятностью. $p(y|x)$ — это так называемое правдоподобие, вероятность данных при условии зафиксированных параметров модели.

Основным недостатком алгоритма Байеса является требование выборки больших размеров для обучения, а также факт, что если исходная условная вероятность равна нулю, предсказанная вероятность также будет равна нулю. Поэтому в последнее время предпочитают использовать другие алгоритмы.

Алгоритм SVD. Представим, что у есть матрица, состоящая из рейтингов (лайков, фактов покупки и т. п.), которые пользователи (строки матрицы) присвоили продуктам (столбцы матрицы). Получается матрица $R = \left(r_{i,q}\right)_{i=1,\,a=1}^{N,M}$, в которой записаны известные нам рейтинги. Как правило, один пользователь не сможет оценить значительную долю продуктов. Поэтому вряд ли будет много продуктов, которые готова оценить значительная доля пользователей. Это означает, что матрица R — сильно разреженная.

Для разреженных матриц обычно используется так называемое сингулярное разложение [1] в виде

$$R = UDV^T.$$

R — матрица большого размера $N \times M$, но малого ранга f (разреженные матрицы часто бывают малого ранга), ее можно разложить в произведение матрицы $N \times f$ и матрицы $f \times M$, тем самым резко сократив число параметров с $N \times M$ до $(N + M)f$.

Основное же свойство SVD заключается в том, что оно дает оптимальное приближение, если в матрице D просто оставить ровно f первых диагональных элементов, а остальные обнулить:

$$X = UDV^{\mathrm{T}} = U\begin{pmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & \sigma_k \end{pmatrix} V^{\mathrm{T}} \approx U\begin{bmatrix} \sigma_1 & \dots & 0 & 0 & \dots & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \dots & \sigma_f & 0 & \dots & 0 \\ 0 & \dots & 0 & 0 & \dots & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \dots & 0 & 0 & \dots & 0 \end{bmatrix} V^{\mathrm{T}}.$$

В диагональной матрице D, которая стоит в середине сингулярного разложения, элементы упорядочены по размеру: $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_k$, так что обнулить последние элементы — значит, обнулить наименьшие элементы. А f подбирается, исходя из размера сингулярных значений матрицы, т. е. тех самых диагональных элементов матрицы D: желательно отбрасывать как можно больше самых малых по значению элементов.

В случае рекомендательных систем получается, что мы представляем каждого пользователя вектором из f факторов U_i, а каждый продукт — вектором из f факторов V_j. Далее, чтобы предсказать рейтинг пользователя i товару j, берем их скалярное произведение $U_i V_j = U_i^T V_j$.

Перед нами стоит задача: по известным оценкам известных продуктов предсказать, насколько хорошо будет оценен новым пользователем каждый продукт.

Введем так называемые базовые предикторы $b_{i,a}$, которые складываются из базовых предикторов отдельных пользователей b_i, и отдельных продуктов b_a, а также просто общего среднего рейтинга по базе μ:

$$b_{i,a} = \mu + b_i + b_a,$$

где μ — средний рейтинг по базе; b_i — средний рейтинг каждого i пользователя; b_a — средний рейтинг каждого a продукта.

Для определения только базовых предикторов необходимо найти такие μ, b_i, b_a, для которых $b_{i,a}$ лучше всего приближают имеющиеся рейтинги. Затем можно будет добавить собственно факторы. Поскольку теперь, когда сделана поправка на базовые предикторы, остатки будут сравнимы между собой, вполне возможно будет получить разумные значения для факторов:

$$\dot{r}_{i,a} = \mu + b_i + b_a + v_a^{\mathrm{T}} u_i, \qquad (1)$$

где v_a — вектор факторов, представляющий продукт a; u_i — вектор факторов, представляющий пользователя i.

Теперь можно вернуться к исходной задаче и сформулировать ее точно: нужно найти наилучшие предикторы, которые приближают величину $\dot{r}_{i,a}$.

Лучшими будут те предикторы, которые дают минимальную ошибку, определяемую следующим образом:

$$L(\mu, b_i, b_a, v_a, u_i) = \sum_{(i,a)\in D} \left(r_{i,a} - \dot{r}_{i,a}\right)^2 = \sum_{(i,a)\in D} \left(r_{i,a} - \mu - b_i - b_a - v_a^T u_i\right)^2.$$

Функцию $L(\mu, b_i, b_a, v_a, u_i)$ минимизируем градиентным спуском: берем частные производные по каждому аргументу и двигаемся в сторону, обратную направлению этих частных производных.

Для компенсирования эффекта переобучения добавляется параметр регуляризации. Иными словами, накладывается штраф за слишком большие значения обучаемых переменных. Например, можно просто добавить в функцию ошибки сумму квадратов всех факторов и предикторов. В результате функция ошибки выглядит как

$$b_*, q_*, p_* = \arg\min_{b,p,g} \sum_{(i,a)} \left(r_{i,a} - \mu - b_i - b_a - q_a^T p_i\right)^2 + \\ + \lambda\left(\sum_i b_i^2 + \sum_a b_a^2 + \|q_a\|^2 + \|p_i\|^2\right), \quad (2)$$

где λ — параметр регуляризации.

Если взять у функции ошибки в формуле (2) частные производные по каждой из оптимизируемых переменных, получим простые правила для градиентного (стохастического) спуска:

$$b_i = b_i + \gamma\left(e_{i,a} - \lambda b_i\right),$$
$$b_a = b_a + \gamma\left(e_{i,a} - \lambda b_a\right),$$
$$q_{a,j} = q_{a,j} + \gamma\left(e_{i,a} p_{i,j} - \lambda q_{a,j}\right),$$
$$p_{i,j} = p_{i,j} = \left(e_{i,a} p_{i,j} - \lambda p_{i,j}\right)$$

для всех j, где $e_{i,a} = r_{i,a} - \dot{r}_{i,a}$ — ошибка на данном тестовом примере, а γ — скорость обучения. Эта модель называется SVD++.

Рассмотрим пример использования описанного выше алгоритма для следующего случая:

- количество пользователей $i = 50$;
- количество категорий предпочтения (товар, категории статей, жанр литературы и т.д.) $a = 10$;
- степень сингулярного разложения $j = 2$;
- матрица рейтингов создается с помощью генератора случайных чисел, которые могут принимать следующие значения:
  - 0 — категория предпочтений не просматривалась;
  - 1 — категория предпочтений просматривалась, но не вызвала ответной реакции пользователя;
  - 2 — категория предпочтений просматривалась и вызвала ответную реакцию пользователя.

В результате выполнения алгоритма получаются следующие результаты для всех пользователей по всем категориям:

- вектор базовых предикатов пользователей (b_i);
- вектор базовых предикатов категорий (b_a);
- матрица векторов факторов категорий ($q_{a,j}$);
- матрица векторов факторов пользователей ($p_{i,j}$);
- средний рейтинг по базе (μ).

Значения получаемых базовых предикатов зависят от «доброты» пользователя: например, первый пользователь, выставляющий максимальные оценки, имеет максимальное значение предиката.

Приведем для сравнения предсказанное значение рейтингов категорий для первого пользователя:
[1.2605181590507217, 0.39919251441107484, –0.46637175802611397, 0.3358525032572892, –0.4658708361423298, 0.35638293991233283, –0.5730704485042799, –0.5692220175436897, 0.26389365744636506, –0.5832468489643301].

Следует обратить внимание на отрицательные значения некоторых рейтингов. Они связаны с погрешностью данного алгоритма, обусловленной относительно небольшим количеством исходной информации. Фактически эти значения соответствуют нулевым рейтингам.

Алгоритм RNSA(The Refined Neighbor Selection Algorithm). С помощью алгоритма кластеризации *K*-средних создается *K* кластеров, каждый из которых состоит из клиентов, имеющих аналогичные предпочтения между собой. Вначале выбирается один произвольный клиент, а в качестве начальной точки центра кластера — k. Тогда каждому клиенту присваивается кластер таким образом, что расстояние между клиентом и центром кластера является максимальным. Коэффициент корреляции Пирсона можно использовать в качестве расстояния:

$$w_{u,a} = \frac{\sum_{i \in I} \left(r_{a,i} - \overline{r}_a\right)\left(r_{u,i} - \overline{r}_u\right)}{\sqrt{\sum_{i \in I} \left(r_{a,i} - \overline{r}_a\right)^2 \sum_{i \in I} \left(r_{u,i} - \overline{r}_u\right)^2}}, \qquad (3)$$

где $w_{u,a}$ — мера близости пользователей a и u; I — множество объектов, оцененные как пользователем a, так и пользователем u; $r_{u,i}$, $r_{a,i}$ — оценка объекта i пользователями u и a соответственно; r_a, r_u — средняя оценка пользователей a и u соответственно

После завершения кластеризации выбирается кластер с наивысшим значением коэффициента корреляции Пирсона от его центра к тестовому клиенту. Прогноз просчитывается для всех клиентов в выбранном кластере.

Ниже представлена последовательность этапов алгоритма RNSA [5]:

Вход: тест-клиент *t*, входной набор данных *S.*

Выход: Соседи.

1. Создать *K* кластеров из *S* методом кластеризации *K*-средних.
2. Найти лучший кластер *C* для *t*.
3. Добавить *t* в лучший кластер *C* и рассматривать его как *v*.
4. Добавить *v* в список соседей.

5. Если число соседей достаточно, возвращаем список соседей. В противном случае, извлекаем v из C, обходим вершины (клиентов) поиском в ширину. Схожесть клиента проверяется по критерию Пирсона (3), если она либо выше, чем нижнее пороговое значение δh, либо ниже, чем верхний порог δl, то v = клиент. Перейти к шагу 4.

На шаге 5 алгоритм завершает свою работу, если число найденных клиентов при поиске в ширину, больше какого-то фиксированного значения. Эта величина может быть определена путем различных экспериментов. Клиент t в список возвращаемых соседей не входит. Формула расчета прогноза:

$$P_{a,i} = A\left(\overline{r_{a,l}}\right) + \frac{\sum_k \left(w_{a,k} \left(r_{k,i} - A\left(\overline{r_{k,i}}\right)\right)\right)}{\sum_k \left| w_{a,k} \right|}, \qquad (4)$$

где $P_{a,\,i}$ — предсказание оценки; $w_{a,\,k}$ — мера близости между пользователями a и k; $A\left(\overline{r_{a,l}}\right)$, $A\left(\overline{r_{k,i}}\right)$ — средняя оценка пользователей a и k соответственно.

Применим данный алгоритм к матрице рейтингов, которая использовалась, в вышеописанном алгоритме SVD++. После выполнения алгоритма с учетом суммирования по формуле (1) получаем вектор предпочтений для первого пользователя по каждой категории:
[0.4047971890674755, 1.3933094711397889, 1.0862712235070604, 0.6473405846451133, 1.263503813085462, 0.741462334544462, 0.7584886141058054, 1.0364811015987234, 1.3698910207764898, 1.2984546475296193].

Как видно из приведенных выше результатов, отсутствуют рейтинги с отрицательными значениями, что свидетельствует о более высокой точности алгоритма RNSA по сравнению с SVD++.

Сравнение алгоритмов. Исходя из полученных результатов, можно сделать следующие выводы о работе алгоритмов SVD++ и RNSA:

по количеству исходных данных — в случае небольшого количества данных для обучения алгоритм SVD++ иногда выдает отрицательные результаты из-за накопленных ошибок в процессе обучения. При использовании алгоритма RNSA даже в случае использования небольших наборов данных для обучения получаются удовлетворительные результаты;

по времени работы — время работы алгоритма SVD++ равно $9.225069444444445 \cdot 10^{-6}$ секунд, время работы алгоритма RNSA составляет $3.525664351851852 \cdot 10^{-5}$ секунд. Очевидно, что алгоритм SVD++ быстрее RNSA. При этом RNSA рассчитывается для каждого пользователя в отдельности, а SVD++ дает результат сразу по всем имеющимся пользователям;

по суммарной погрешности — погрешность алгоритма SVD++ 8.050711251156905, погрешность алгоритма RNSA 4.0.

Исходя из полученных результатов рекомендуется использовать алгоритм RNSA в случае небольшого количество данных и для получения более точных результатов. Если база клиентов и товаров имеют большие размеры, рекомендуется использовать алгоритм SVD++.

ЛИТЕРАТУРА

[1] Berry M.W. Large scale singular value computations. *International Journal of Supercomputer Applications*, 1992, no. 6(1), pp. 13–49.

[2] Billsus D., Pazzani M.J. *Learning Collaborative Information Filters*.

[3] Wikipedia, *Статья о коллаборативной фильтрации*. http://ru.wikipedia.org/wiki/

[4] MachineLearning, *Статья о байесовском классификаторе*. http://www.machinelearning.ru/wiki/

[5] Taek-Hun Kim, Sung-Bong Yang. *An Effective Threshold-based Neighbor Selection in Collaborative Filtering. Dept. of Computer Science*, Yonsei University, Seoul, pp. 10–749.

Статья поступила в редакцию 10.06.2013

Ссылку на эту статью просим оформлять следующим образом:

Королева Д.Е., Филиппов М.В. Анализ алгоритмов обучения коллаборативных рекомендательных систем. *Инженерный журнал: наука и инновации*, 2013, вып. 6. URL: http://engjournal.ru/catalog/it/hidden/816.html

Королева Дарья Евгеньевна — студентка 2-го года магистратуры кафедры «Программное обеспечение ЭВМ и информационные технологии» МГТУ им. Н.Э. Баумана. Область научных интересов: машинное обучение. e-mail: zireaell@land.ru

Филиппов Михаил Владимирович родился в 1953 г., окончил МИФИ в 1977 г. Канд. техн. наук, доцент кафедры «Программное обеспечение ЭВМ и информационные технологии» МГТУ им. Н.Э. Баумана. Автор более 50 научных и учебно-методических публикаций в области автоматизированного проектирования и цифровой обработки сигналов Область научных интересов: цифровая обработка сигналов, распознавание образов, разработка средств защиты информации. e-mail: profitbig@rambler.ru